Performance Analysis of Secure Web Server Based on SSL

Xiaodong Lin[1], Johnny W. Wong[1], and Weidong Kou[2]

[1] Department of Computer Science,
University of Waterloo, Waterloo, Ontario, Canada N2L 3G1
{xdlin,jwwong}@bbcr.uwaterloo.ca
[2] E-Business Technology Institute,
University of Hong Kong, Pokfulam Road, Hong Kong
wdkou@eti.hku.hk

Abstract. In recent years, protocols have been developed to ensure secure communications over the Internet, e.g., the secure sockets layer (SSL) and secure electronic transaction (SET). Deployment of these protocols incurs additional resource requirements at the client and server. This may have a negative impact on system performance. In this paper, we consider a scenario where users request information pages stored on a web server, and some of the requests require secure communication. An analytic model is developed to study the performance of a web server based on SSL. In our model, the details of the client-server interactions found in a typical SSL session are represented explicitly. Input parameters to this model are obtained by measuring an existing SSL implementation. Numerical examples on the performance characteristics of SSL are presented.

1 Introduction

In recent years, we have witnessed a general acceptance of the Internet by businesses and consumers. A key requirement to the success of Internet business is secure communication. It is generally known that messages sent on the Internet are subject to three types of security threats, namely eavesdropping, modification, and impersonation [1]. Trusted security mechanism and protocols have been developed to ensure secure communications over the Internet. An important security protocol is the secure sockets layer (SSL) [2,3]. It has been implemented in all the major web browsers and in web servers like Apache [4], Lotus Domino server [5] and IBM HTTP server [6]. Another important security protocol is secure electronic transaction (SET) [7].

Deployment of security protocols, such as SSL and SET, incurs additional resource requirements at the client and server. This may have a negative impact on system performance, e.g., increased response time. Performance evaluation of SSL, SET or other security mechanisms, has not received much attention until recently. In [8], the performance improvement of SSL when caching of session keys is used, is evaluated. In another study [9], performance results for a transport layer security protocol are presented. That study was based on the use of a benchmark tool SPECweb96 [10]. In addition, a performance comparison of SSL and SET is reported in [11].

Our work is different from those in [8,9,11] in the sense that analytic modeling is used to study the performance of SSL. More specifically, we consider a scenario where users request information pages stored on a web server, and some of the requests require secure communication. A model is developed to represent the details of client-server interactions during a typical SSL session. As part of our investigation, we obtain input parameters to our model by measuring an existing SSL implementation. Analytic results are then obtained; these results are used to evaluate the performance penalty incurred by SSL. The issue of scalability is also investigated. The advantages of our model include (i) the effects of implementation are represented by the actual measurements and thus the model does not have to consider implementation details, and (ii) the mean response time results are valid for arbitrary distribution patterns of user think time and the various service time parameters.

The rest of this paper is organized as follows. In section 2, the details of a typical SSL session are described with a view to capturing the client-server interactions. Our model is described in section 3. Exact analytic results for the mean response time are derived. System measurement techniques to obtain input parameters are also described. In section 4, numerical examples showing the performance characteristics of a secure web server based on SSL are presented. These examples are designed to show the impact of the number of users, the fraction of user requests that require secure communication, the type of cryptographic algorithms used, and the size of the html file retrieved, on response time performance. Finally, section 5 contains a summary of our findings and some concluding remarks.

2 Secure Sockets Layer Protocol Session

Secure Sockets Layer (SSL) protocol is designed to provide secure communication. It performs server authentication, and optionally client authentication. With SSL, private information is protected through encryption, and a user is ensured through server authentication that he/she is communicating with the desired web site and not with some bogus web site. In addition, SSL provides data integrity, i.e., protection against any attempt to modify the data transferred during a communication session.

Our investigation of SSL is based on a publicly available implementation called SSLeay [12]. This would allow us to collect measurement data that can be used to characterize the input parameters. Our measurement experiments are based on SSLeay version 0.6.6b, which is an implementation of SSL v2. This version is selected because it is a stable implementation and is therefore a good candidate to illustrate our modeling approach. Our model can easily be extended to study the behavior of later versions of SSL (v3 and transport layer security [13]).

For convenience, we consider the case where an SSL session is set up to request a web page (or https). To facilitate the development of our model, we show in Figure 1 the events and activities as seen by the web server. The SSL session starts at the end of a user think time, which is indicated by the reception of a TCP connection request on port 443 - the default port number assigned to https. The web server then proceeds with setting up the TCP connection, and this activity ends when the TCP connection is established. The web server then waits for a "client hello" message. When such a message is received, the server parses the message, and prepares a "server hello" message for transmission back to the client.

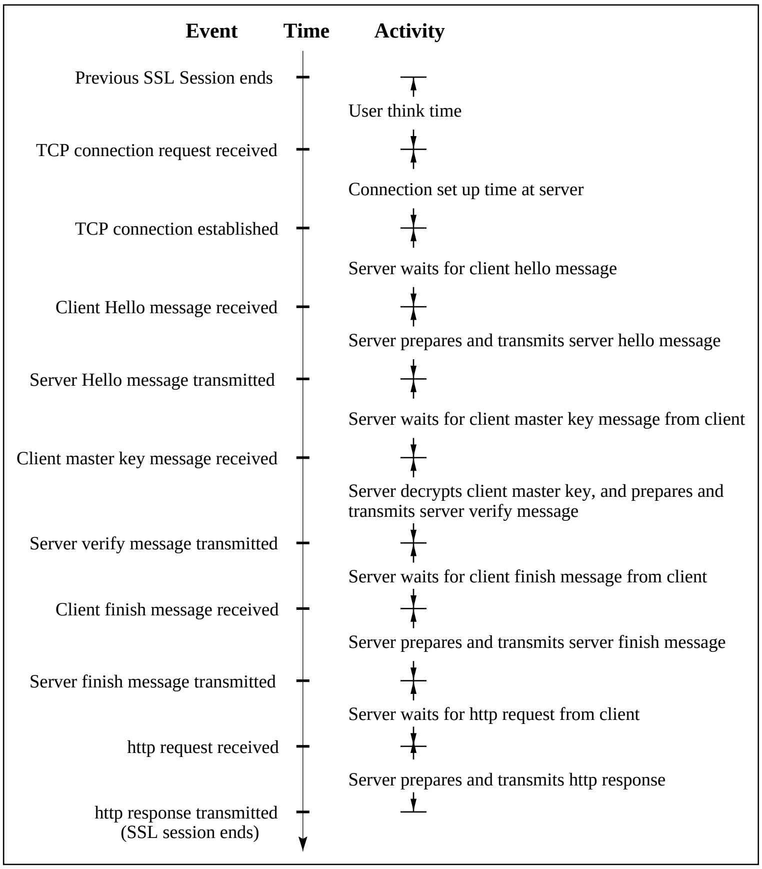
Event
Time
Activity
Previous SSL Session ends
User think time
TCP connection request received
Connection set up time at server
TCP connection established
Server waits for client hello message
Client Hello message received
Server prepares and transmits server hello message
Server Hello message transmitted
Server waits for client master key message from client
Client master key message received
Server decrypts client master key, and prepares and transmits server verify message
Server verify message transmitted
Server waits for client finish message from client
Client finish message received
Server prepares and transmits server finish message
Server finish message transmitted
Server waits for http request from client
http request received
Server prepares and transmits http response
http response transmitted (SSL session ends)

Fig. 1. SSL session

After the "server hello" message has been transmitted, the web server waits for a "client master key" message from the client. Upon receiving this message, the web server decrypts the client master key, and prepares a "server verify" message and transmits it to the client. At end of transmission, the server waits for a "client finish" message. When this message is received, the server prepares a "server finish" message for transmission to the client. The SSL handshake ends when this message is transmitted.

The web server next waits for an http request from the client. Upon receiving this request, the web server prepares an http response. When this response is transmitted, the SSL session ends and the user starts the next think time.

Similarly, we show in Figure 2 the events and activities for the case where secure communication is not required, i.e., a normal http request.

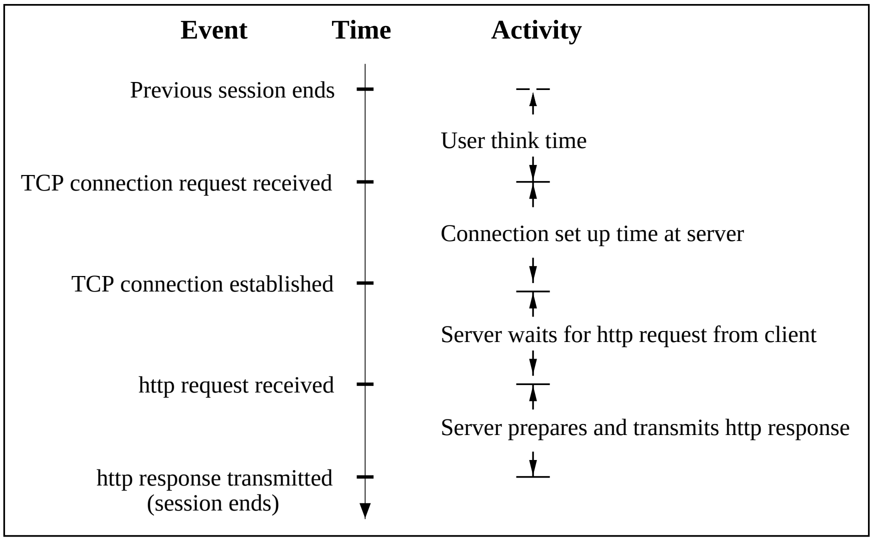
Event
Time
Activity
Previous session ends
User think time
TCP connection request received
Connection set up time at server
TCP connection established
Server waits for http request from client
http request received
Server prepares and transmits http response
http response transmitted (session ends)

Fig. 2. Normal http request

3 Performance Model and System Measurement

3.1 Performance Model

Our performance model is a closed queueing network representing N users requesting web pages from a web server (see Figure 3). There are two types of requests: type 1 requires secure communication and type 2 is normal http. A request is generated at the end of a user think time. This request is of type 1 with probability p and of type 2 with probability $1 - p$.

There are two service centres: the web server and a "delay" server. The delay server represents all web server activities while waiting for the client, and any network delays. Each type 1 request cycles through the web server and delay server a number of times, following the activities shown in Figure 1. As a result,

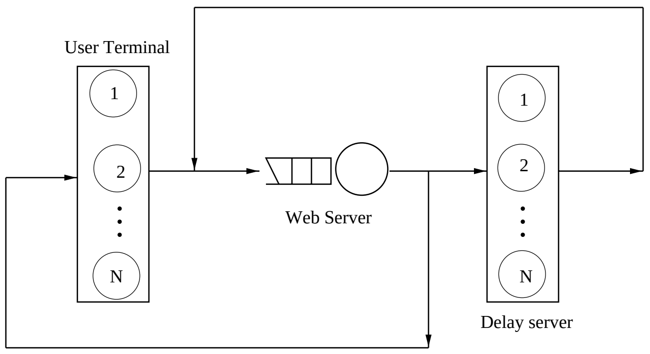

Fig. 3. Performance model of secure communication sessions

a type 1 request visits the web server several times, as shown by the stages of service in Figure 4. The type 1 request also visits the delay server a number of times, shown by the stages in Figure 5. After visiting the web server for the last time (which corresponds to the activity "server prepares and transmits http response"), a type 1 request is complete, i.e., the http response is returned to the user, and the user starts the next think time. Similarly, for type 2 requests, the stages of service at the web server and delay server are shown in Figures 6 and 7 respectively.

In general, processing of a user request requires usage of resources such as processor, memory and database servers. For convenience, we assume that the required resources are approximated by a number of service time parameters, one for each of the web server or delay server stages. Furthermore, the web server is modeled by a single server queue with processor-sharing discipline [14]. This discipline provides fair sharing of resources among all outstanding user requests. The user terminal is modeled as an "infinite server", or no queueing. This is a well-accepted model for interactive systems. The delay server is also modeled as an "infinite server". This assumption can be justified as follows. The time spent by a request at the delay server has two components: processing at the client and network delay. Each client is assumed to be running on a dedicated machine, and hence, there is no queueing at the client. As to the network delay, we assume that the time spent at the server machine to transmit or receive packets is small compared to that required for SSL protocol processing, similarly for time spent in the transport network. The no queueing assumption can therefore be used for the network delay component also.

For our queueing network model, the input parameters are the number of users N, the mean think time h, the fraction of user requests that require secure communication p, and the mean service time of each stage at the web server and delay server for each of the two types of requests (as shown in Figures 4 to 7).

Stage of service at web server	Mean service time (measured)
Connection set up time at server	0.406 msec
Server prepares and transmits server hello message	0.480 msec
Server decrypts client master key, and prepares and transmits server verify message	11.815 msec
Server prepares and transmits server finish message	0.278 msec
Server prepares and transmits HTTP response	1.937 msec

Fig. 4. Stages of service at web server for type 1 requests

Stage of service at delay server	Mean service time (measured)
Server waits for client hello message	2.536 msec
Server waits for client master key message	12.407 msec
Server waits for client finish message	0.472 msec
Server waits for http request	3.921 msec

Fig. 5. Stages of service at delay server for type 1 requests

Stage of service at web server	Mean service time (measured)
Connection set up time at server	0.322 msec
Server prepares and transmits http response	1.911 msec

Fig. 6. Stages of service at web server for type 2 requests

3.2 Analytic Results

Our model belongs to the types of queueing network models analyzed in [14]. Specifically, each web server or delay server stage is represented by a seprate customer class, and the class change feature allows us to model the customer routing between the web server and delay server, as characterized by the ordering of activities shown in Figure 1. The results from [14] are therefore directly applicable.

Stage of service at delay server	Mean service time (measured)
Server waits for http request	0.094 msec

Fig. 7. Stage of service at delay server for type 2 requests

For our investigation, the state description of interest is $S = (n_0, (n_{11}, n_{12}), (n_{21}, n_{22}))$ where n_0 is the number of users in the thinking state, and n_{1r} and n_{2r} are the number of type r requests at the web server and the delay server respectively, $r = 1$, 2. Using the results in [14], we obtain $P(S)$, the steady state probability that the system is in state S. From $P(S)$, one can readily obtain the mean values for n_0, n_{11}, n_{12}, n_{21} and n_{22} (denoted by $\overline{n}_0$, $\overline{n}_{11}$, $\overline{n}_{12}$, $\overline{n}_{21}$ and $\overline{n}_{22}$ respectively).

Let $\overline{T}_r$ be the mean response time of type r requests, we use Little's formula [15] to obtain:

$$\overline{T}_1 = \frac{(\overline{n}_{11} + \overline{n}_{21})h}{\overline{n}_0 p} \tag{1}$$

and

$$\overline{T}_2 = \frac{(\overline{n}_{12} + \overline{n}_{22})h}{\overline{n}_0(1-p)} \tag{2}$$

Finally, the mean response time over all requests is

$$\overline{T} = \frac{(N - \overline{n}_0)h}{\overline{n}_0} \tag{3}$$

It should be noted that response time is measured from the beginning to end of an SSL session (or a normal http session). It includes the delays while waiting for the client process, and any network delays.

It should also be noted that the formulas for the mean response time ($\overline{T}_1$, $\overline{T}_2$ and $\overline{T}$) are valid for a wide range of distributional assumptions for the think time, and for the service times at the various web server and delay server stages. The reason is that the steady state probability $P(S)$ depends only on the mean think time and the mean service times at these stages [14].

3.3 System Measurement

In this subsection, we discuss our methodology to obtain the mean service time for each of the web server and delay server stages. We built a web server based on SSLeay [12]. The server is written in C and compiled with gcc 2.6 with optimization. Our experimental system consists of a Sun Ultra 10 running SunOS 5.6, which works as the web server. The client machine is a Sun SPARCstation

10 running SunOS 5.5. These two machines are connected to an 100 Mbit/s Ethernet. The required service time parameters can be determined if we know the event times of all the events in Figures 1 and 2. This can be accomplished if we are able to measure the time at which a message is received at the server, and the time at which transmission of a message at the server is finished.

To measure the time at which a message is received at the server, we use the I/O multiplexing model of Unix network I/O [16]. This model allows us to determine exactly when data for a given message are received. We thus modify the available SSLeay source code and place a "select" system call before every server subroutine that attempts to receive data from client. Each time a message is received, the event time is obtained by using the "gettimeofday" system call which returns the current time at a resolution of microseconds.

Measuring the time at which transmission of a message at the server is finished is quite straight-forward. One simply uses the "gettimeofday" system call to get the event time when transmission is complete.

3.4 Measurement Results

We conducted two measurement experiments to obtain values for the service time parameters. In these experiments, we used a scenario of only one user interacting with a web server. The html file size is 1.0 Kbyte. Public-key cryptography is based on 512-bit RSA [17], and secret-key encryption is based on RC4 [17].

Data were collected under the condition of no other applications running on the server and client machines. The first experiment involves SSL sessions (or type 1 requests) only, while the second experiment is concerned only with normal http (or type 2 requests). The measured values of mean service time at the various web server and delay server stages are included in Figures 4 to 7.

4 Numerical Examples

For closed queueing networks, the amount of computation required to obtain numerical results for performance measures such as mean response time may be substantial. Efficient computational algorithms are widely available. For our model, numerical results are obtained by using the algorithm reported in [18].

We first investigate the effect of the number of users N and the parameter p on the mean response time over all requests. The results for $p = 0.1$, 0.2 and 0.3, are shown in Figure 8. The mean think time h is assumed to be 2 seconds.[1] It is observed that with a larger p, there is a performance penalty (i.e., a larger mean response time) even when N is small. The incremental mean response time increases with N and p. The performance penalty incurred by SSL can be explained as follows.

[1] This value of h is selected such that the amount of computation is not excessive for reasonably large values of N. For a larger h, the performance characteristics are similar, except the system is able to support more users.

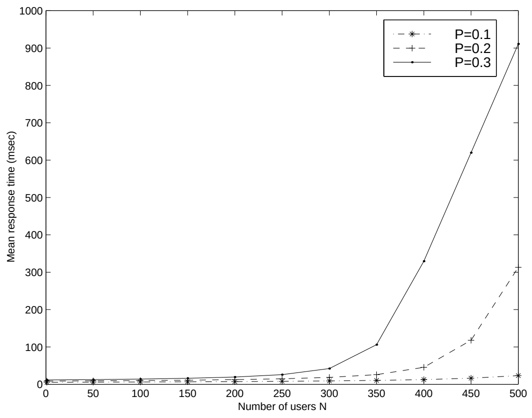

Fig. 8. Mean response time over all requests

Under SSL, significant processing time is required at the web server to decrypt the client master key because public-key cryptography is used. As shown in Figure 4, the mean service time for this activity is 11.815 msec compared to 1.937 msec for preparing an http response. Also, in SSL, secret-key cryptography is used when the web server prepares (i) the "server finish" message and (ii) the http response. This would add to the service times for the SSL session. Further delays are incurred because cryptographic hash functions are computed for all server activities except TCP connection establishment. In addition, an SSL session has more client-server interactions than the normal http. This is a result of the handshake protocol required to set up the SSL session. These additional interactions will incur delay, not only because of the processing required, but also the need to wait for network round-trip time, and any processing required at the client.

To further illustrate the response time performance of SSL, we plot in Figure 9 and 10 the mean response time of type 1 and type 2 requests respectively. It is observed that an increase in p results in performance degradation for both types of requests.

Our second set of results are concerned with the impact of increasing the security strength on SSL performance. Consider first public-key cryptography. The results in Figure 8 are based on measured data for an 512-bit RSA. The longer the RSA key size, the more secure is the SSL protocol. We measured the processing time required on a Sun Ultra 10 for three different RSA private key sizes, and the results are shown in Table 1. Numerical results for mean response time are then obtained for these three key sizes and for $p = 0.2$. These results are shown in Figure 11. We observe that the response time performance degrades significantly as the RSA key size increases. The reason is that RSA private key cryptographic operation incurs a significant overhead at the web server. The response time can be improved by using an RSA public key cryptography accelerator that is implemented in hardware, e.g., CryptoSwift II [19].

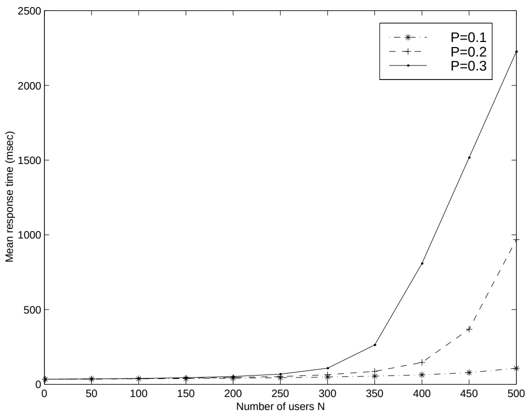

Fig. 9. Mean response time of type 1 requests

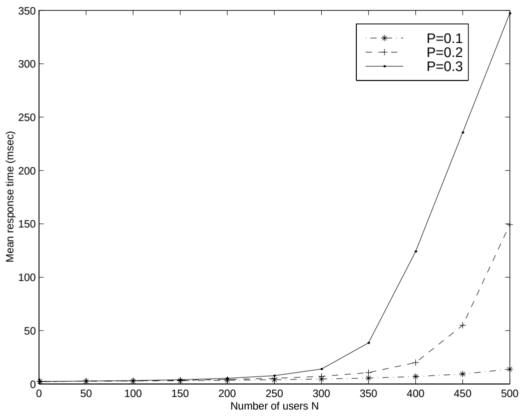

Fig. 10. Mean response time of type 2 requests

Table 1. RSA private key cryptographic operation performance

RSA Key Size	Performance
512 bits	11 msec
1024 bits	68 msec
2048 bits	481 msec

We next consider the performance impact of secret-key encryption. The results in Figure 8 are based on measured data for RC4. When a higher level of security is required, recommended encryption algorithms include IDEA [17] and Triple DES [17]. We measured the encryption rate on a Sun Ultra 10 for these three algorithms, and the results are shown in Table 2. Numerical results for mean response time are then obtained for these three cases, and for $p = 0.2$.

These results are shown in Figure 12. We observe that the mean response time increases as stronger secret-key encryption algorithms are used. The increase is mainly due to the overhead incurred by secret-key cryptographic operation when applied to the html file. Compared to the results in Figure 11, we observe that for the parameters under consideration, the performance penalty is small compared to that incurred by public-key cryptography. However, the observation may be different if the html file size is much larger. This is due to the higher overhead incurred by secret-key cryptography, resulting in a significant increase in the mean response time. To illustrate this point, we show in Figure 13 the mean response time performance when the html file sizes are 1 and 2 Mbytes instead of 1 Kbyte. Compared to the results in Figure 11, the performance penalty resulting from secret-key encryption is comparable to that from public-key cryptography.

Table 2. Performance of RC4, IDEA and Triple DES

Encryption Algorithm	Encryption Speed
RC4	10568.51 Kbyte/sec
IDEA	2794.01 Kbyte/sec
Triple DES	809.23 Kbyte/sec

5 Conclusions

We have presented an analytic model for studying the performance of SSL. Our model is based on a detailed representation of the client-server interactions resulting from an SSL session. Numerical results showing the tradeoff between security and response time performance have been obtained. These results provide useful insight into the performance characteristics of a secure web server under a range of operating conditions.

Our analytic model is rather general because the results are valid for arbitrary think time and arbitrary service time distributions at the web server and delay server. In addition, our modeling approach can readily be extended to study other secure communication protocols, e.g., SET.

Acknowledgements. This work was supported by the Natural Sciences and Engineering Research Council of Canada. The authors would also like to thank the anonymous reviewers for their constructive comments.

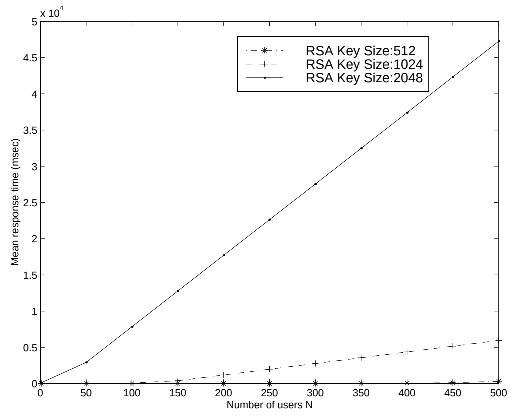

Fig. 11. Mean response time for different RSA private key sizes

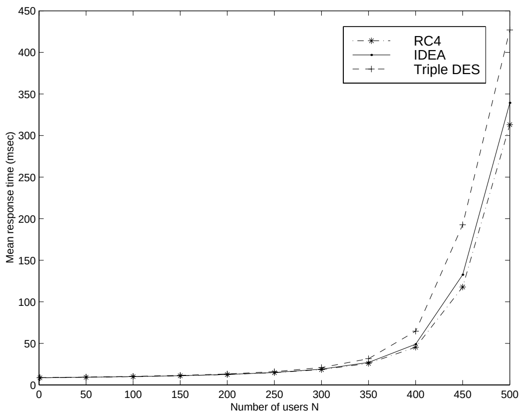

Fig. 12. Mean response time for different symmetric key encryption algorithms

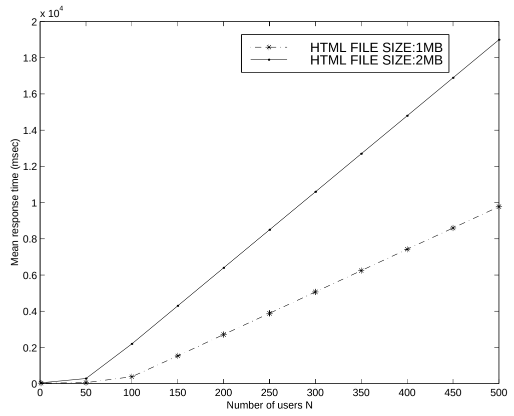

Fig. 13. Mean response time for different html file sizes

References

1. Gregory B. White, Eric A. Fisch, Udo W. Pooch. Computer System and Network Security. CRC Press, Inc., New York, 1996.
2. Kipp E.B. Hickman. SSL 2.0 Protocol Specification. February 1995. http://www.netscape.com/eng/security/SSL_2.html
3. Alan O. Freier, Philip Karlton, Paul C. Kocher. SSL 3.0 Protocol Specification. March 1996. http://home.netscape.com/eng/ssl3/index.html
4. Apache Server. http://www.apache.org
5. Lotus Domino Server. http://www.lotus.com/home.nsf/welcome/domino
6. IBM HTTP Server. http://www.as400.ibm.com/products/http/httpindex.htm
7. MasterCard International Incorporated, Visa International. The SET Specification 1.0, Dec. 1997. http://www.setco.org/set_specifications.html
8. Arthur Goldberg, Robert Buff, Andrew Schmitt. Secure Web Server Performance Dramatically Improved by Caching SSL Session Keys. Workshop on Internet Server Performance, SIGMETRICS '98, Madison, Wisconsin, June 1998.
9. George Apostolopoulos, Vinod Peris, Debanjan Saha. Transport Layer Security: How much does it really cost? Proc. INFOCOM '99, New York, March 1999.
10. The Standard Performance Evaluation Corporation. SPECweb96 Benchmark, 1996. http://www.spec.org/osg/web96/
11. Chris Le Tocq, Steve Young. Set Comparative Performance Analysis: Gartner Group White Paper. http://www.setco.org/download/setco6.pdf
12. T.J. Hudson, E.A. Young. SSLeay Programmer Reference. January 1996. http://www.psy.uq.oz.au/~ftp/Crypto/ssl.html
13. T. Dierks, C. Allen. RFC2246: The TLS Protocol Version 1.0, January 1999.
14. F. Baskett, K.M. Chandy, R.R. Muntz, F.G. Palacios. Open, Closed and Mixed Network of Queues with Different Classes of Customers. Journal of the ACM 22(2), April 1975, 248-260.
15. J.D. Little. A Proof of the Queueing Formula $L = \lambda W$. Operations Research 9(3), 1961, 383-387.
16. W. Richard Stevens. UNIX Network Programming, Volume 1, Second Edition: Networking APIs: Sockets and XTI. Prentice Hall, Upper Saddle River, New Jersey, 1998.
17. A.J. Menezes, P.C. van Oorschot, S.A. Vanstone. Handbook of Applied Cryptography. CRC Press, New York, 1997.
18. J.W. Wong. Queueing Network Models for Computer Systems. Ph.D. thesis, University of California at Los Angeles, 1975.
19. Rainbow Technologies Company. Secure Web Server and VPN (IPSec) Acceleration. http://isg.rainbow.com/index.html
20. Special Issue on Web Performance, IEEE Network 14(3), May/June 2000.
21. The Standard Performance Evaluation Corporation. SPECweb99 Benchmark, 1999. http://www.spec.org/osg/web99/
22. R. Hariharan, W. K. Ehrlich, D. Cura, P. K. Reeser. End to End Performance Modeling of Web Server Architectures. Performance Evaluation Review 28(2) September 2000, 57-63.